Plan for Distributed Key-Value Store using RPC/RDMA

---

1. Project Structure

- **Components**:
  1. Shared map on Shared memory implemented using Boost C++ libraries.
  2. KV Manager with RPC implementations (Server, Client, Handles/Engine).
  3. RDMA integration for remote communication.
  4. Control plane with Hash function to distribute the data to compute nodes.
  5. CMake files for building and compiling the project.

- **Directory structure**:
  1. /src: Contains source code files with the components mentioned above.
  2. /include: Includes necessary base class implementations or libraries.
  3. /common: Configuration files, system or any general purpose files.
  4. /build: Used for the executables after compilation using CMake.
  5. /docs: For detailed documentation of the project.
  6. /scripts: Contains scripts used to run the processes.

---

2. Local and Remote Access Implementation

- **Local Access:**
  1. Implementation of classes that allow clients to access data directly from the shared map.
  2. Shared memory access should provide efficient read/write operations for the process running on the local node.
- **Remote Access:**
  1. Implementation of RDMA procedures with RPC to handle remote access between nodes using KV manager.
  2. Set up of the necessary RDMA logic and protocols to call relevant server/client operations remotely.
  3. Ensuring that the system supports seamless remote access with minimal latency using RDMA.

---

3. Build and Configuration

- **CMake Setup**:
  1. Using CMakeLists.txt files in respective directories to facilitate the build process.
  2. Maintaining a Build folder to have all the executables after compilation.
  3. CMake for managing build dependencies and ensuring cross-platform compatibility.

---

4. GitHub Workflow

- **Version Control**: Maintaining GitHub repository with branching workflows.
    1. Main branch: Only stable code is merged here after testing.
    2. Feature branches: For experimenting or developing new features.
